CAUCHY – Jurnal Matematika Murni dan Aplikasi
Volume 11 (1) (2026), Pages 524-539
p-ISSN: 2086-0382; e-ISSN: 2477-3344

Evaluating Hierarchical Pathfinding A* (HPA*) for Multi-Order Routing in a Small Warehouse Layout

Ig. Prasetya Dwi Wibawa[1], Meta Kallista[2*], Ramdhan Nugraha[1], Heni Widayani[3], Harish Chandra Bhandari[4], and Angga Rusdinar[1]

[1]*Electrical Engineering, School of Electrical Engineering, Telkom University, Indonesia*
[2]*Computer Engineering, School of Electrical Engineering, Telkom University, Indonesia*
[3]*Mathematics Department, Universitas Islam Negeri Maulana Malik Ibrahim Malang, Indonesia*
[4]*Department of Mathematics, School of Science, Kathmandu University, Nepal*

Abstract

Hierarchical Pathfinding A* (HPA*) is a hierarchical search framework that partitions grid-based environments into clusters to reduce computational time while preserving a near-optimal path. The warehouse layout features cross-aisle connectivity, and multi-order optimization is performed using the HPA* algorithm, which integrates travel times with multi-rack picking times to the objective cost function. We simulate by assigning 30 random orders, with approximately 10000 items stored in the warehouse and 50 item types. The travel time is calculated assuming a picker has a constant speed of 1.2 m/s along edges, the picking time is proportional to the number of items picked per rack, and a small warehouse layout. Estimated cycle times of the orders (travel plus picking time) range from 114.4 to 349.9 seconds using the HPA* optimization, with a mean of 232.0 seconds. From the optimization results, orders require an average of 5.2 rack visits, ensuring that the picker travels more than two racks per order. The HPA* reduces the original low-level graph (50 nodes and 61 edges, including base and stage station) to a graph with 22 nodes and 17 edges, enabling faster route computation while preserving observed cycle-time patterns when combined with picking-time durations. Compared to A*, given the layout and orders, HPA* demonstrates an efficient warehouse path-planning method that reduces search computation while maintaining near-optimal routing performance.

Keywords: HPA*, hierarchical pathfinding, warehouse routing, multi-order, route optimization.

1. Introduction

Path planning in modern warehouses is inherently challenging due to high rack density, long, narrow aisles [1], [2], cross-aisle connectivity [3], and fluctuating traffic arising from human [4], semi-autonomous [5], or fully autonomous pickers, for example, using automated guided vehicles (AGVs) and autonomous mobile robots (AMRs) [6], [7], [8]. Planners must generate collision-free routes that balance travel distance, traversal time, and rack-side service time while respecting safety margins and right-of-way rules [9], [10], [11]. Practical warehouse systems exhibit considerable operational complexity. Multi-order batching, for example, increases the frequency of rack

*Corresponding author. E-mail: metakallista@telkomuniversity.ac.id

Submitted: December 24, 2025 Reviewed: December 31, 2025 Accepted: February 28, 2026
DOI: https://doi.org/10.18860/cauchy.v11i1.39659

visits within a single route [12]. In addition, aisle blockages and transient congestion necessitate rapid re-planning, whereas fleet coordination requires establishing predictable trajectories to mitigate deadlocks and bottlenecks [13], [14]. In dynamic contexts, planning algorithms must be computationally efficient to enable real-time responses and maintain near-optimal performance despite fluctuations in layout configurations and workload intensities over shifts and seasons [15], [16].

A wide range of techniques is employed for pathfinding within warehouses. For multiple orders (or batch picking) in parallel-aisle warehouses, routing and path-planning strategies include grid or graph-based deterministic search, sampling-based motion planning, metaheuristics and evolutionary optimization, learning-based planning, multi-agent path finding (MAPF) and coordination planners, and time-window or reservation-table approaches. Grid-based or graph-based methods discretize the warehouse into nodes and edges and compute shortest routes, e.g., Dijkstra, A⋆, and HPA⋆, making them well-suited to rule-compliant aisle navigation [17], [18]; sampling-based planners such as PRM and RRT/RRT⋆ explore continuous configuration spaces to quickly build feasible paths around racks and obstacles when strict grid models or turning envelopes make graph search less suitable [19], [20]; metaheuristics and evolutionary optimization, i.e., genetic algorithm (GA), ant colony optimization (ACO), and adaptive large neighborhood search (ALNS) treat routing as combinatorial optimization to deliver near-optimal routes under dynamic constraints—recent GA and improved ACO variants reduce runtime and boost robustness, while rolling-horizon ALNS enables rapid re-planning in live warehouse logistics [21], [22], [23]; learning-based planning employs reinforcement learning (RL) or deep models to adapt policies online to layout and traffic changes, achieving real-time decisions in complex maps and shortened travel through data-driven guidance [24], [25]; MAPF-based coordinators compute near-optimal, collision-free routes for an entire robot fleet and remain robust to delays and unexpected obstacles, while still scaling to large 2D/3D warehouses—with effectiveness that can vary based on warehouse layout parameters [15], [26]; and time-window/reservation-table schemes reserve edges in time to keep trajectories predictable and detect conflicts early, enabling online replanning that limits congestion, deadlocks, and bottlenecks. [18], [11], [27].

Hierarchical Pathfinding A⋆ (HPA⋆) is particularly effective in warehouses with parallel and cross-aisles because it groups the grid into clusters and places portals at the entrance and exit points between them. This preserves fine-grained detail where it matters most—at cross-aisle intersections, where route choices are made—while compressing long corridor segments into a smaller abstract graph, thereby reducing time[19]. By partitioning the floor into clusters and exposing inter-cluster portals, HPA⋆ reduces the number of nodes and edges explored during planning, enabling faster computation with near-optimal routes—a critical property when multiple orders and agents share the same infrastructure [28], [29]. Our contribution is a warehouse-specific extension of Hierarchical Pathfinding A⋆ (HPA⋆) that introduces a cross-aisle-aware framework that jointly optimizes multi-order warehouse routes by integrating travel time and rack-side picking time into a single objective and using a shared hierarchical graph to coordinate pickers. The HPA⋆ integrates travel time and picking time into a single cost function, supports multi-order routes that visit multiple racks per cycle, and provides a common topological layer for item pickers. Consequently, HPA⋆ provides an efficient, scalable foundation for autonomous navigation and warehouse operations management without compromising the cycle-time patterns required for throughput optimization. Unlike conventional A⋆ or batching-only approaches, our method compresses the low-level warehouse graph (from 50 nodes/61 edges to 22 nodes/17 edges) while preserving near-optimal cycle-time patterns.

2. Methods: Warehouse Layout and Path Planning Algorithm

This section outlines the end-to-end workflow used to evaluate A⋆ and HPA⋆ in the proposed warehouse setting. We first formalize the warehouse layout as a weighted graph and define a cycle-time cost that combines travel time and rack-level picking time. We then describe the

baseline low-level A⋆ planner, the hierarchical abstraction and refinement steps of HPA⋆, and finally the multi-order objective and computational complexity used for comparison.

2.1. Warehouse Graph and Cost Model

We model the warehouse as a weighted graph

$$G = (V, E, w),$$

where V contains rack nodes including $\{\mathsf{BASE}, \mathsf{STAGE}\}$, E is the set of traversable edges, and $w : E \to \mathbb{R}_{>0}$ denotes travel times. Modeling warehouse routing as a graph (or grid graph) is now common in path-planning research because it makes it straightforward to apply heuristic search methods like A⋆ and its variants on structured warehouse layouts. In practice, A⋆—including hybrid extensions—has been used in warehouse-style environments to shorten travel distance, reduce unnecessary turning, and improve overall time and energy efficiency [30], [31]. For each edge $e \in E$ with line distance d_e, the travel time is

$$w(e) = t_e = \frac{d_e}{s}, \tag{1}$$

with constant item picker speed, s, (in our simulation, we assume the picker speed is constant, i.e., $s \approx 1.20 m/s$). We use speed references from Autonomous Mobile Robots (AMRs) used in warehouses, which operate at common speeds of 1-2 m/s. The adopted speed is consistent with industrial AMR specification sheets (for instance, the MiR1350 AMR lists a maximum speed of 1.2 m/s) and aligns with safety guidance under ISO 3691-4, which mandates speed control for driverless industrial trucks and establishes international standards for mobile robot safety.

Let an order o require a set of rack visits, denoted as $R(o) = \{r_1, \ldots, r_{m(o)}\}$. Define $p : V \to \mathbb{R}_{\geq 0}$ as the rack-level picking time, which is proportional to the number of items retrieved at each stopping rack. This choice indicates that the picking time depends on handling at visited locations, whereas routing policies determine the travel time. Travel time often dominates the picker-to-parts operation, leading to the common practice of modeling total cycle time as the sum of travel time and handling or picking time [32], [33]. For a feasible low-level path π that starts at BASE, visits all racks in $R(o)$, and ends at STAGE, the *cycle time* is

$$\mathrm{CT}(o, \pi) = \underbrace{\sum_{e \in \pi} w(e)}_{\text{travel time}} + \underbrace{\sum_{r \in R(o)} p(r)}_{\text{picking time}}. \tag{2}$$

This travel and picking time-cost structure aligns with the warehouse design presented in Fig. 1, in which cycle time is adopted as the primary performance measure for order fulfillment in the proposed method [34].

2.2. A⋆ on the Low-Level Graph

We solve each waypoint-to-waypoint segment or subpath of the route (e.g., $\mathsf{BASE} \to r_1$, $r_1 \to r_2$, $\ldots$, $r_{m(o)} \to \mathsf{STAGE}$) with A⋆, minimizing $f(n) = g(n) + h(n)$, where:

$$g(n) = \text{accumulated travel time from the subpath's start node to } n,$$
$$h(n) = \frac{\|\mathbf{x}(n) - \mathbf{x}(\text{goal})\|_2}{s}.$$

Here, $\mathbf{x}(n) = \{x_m(n), y_m(n)\} \in \mathbb{R}^2$ denotes the geometric position of a node in 2D node coordinates, where both $x_m(n)$ and $y_m(n)$ are positions measured (in meters). The variable $\mathbf{x}(\text{goal})$ denotes the 2D coordinates of the target node for the current segment, e.g., a rack or STAGE, taken from the warehouse graph, and s is the constant picker speed (in meters/second).

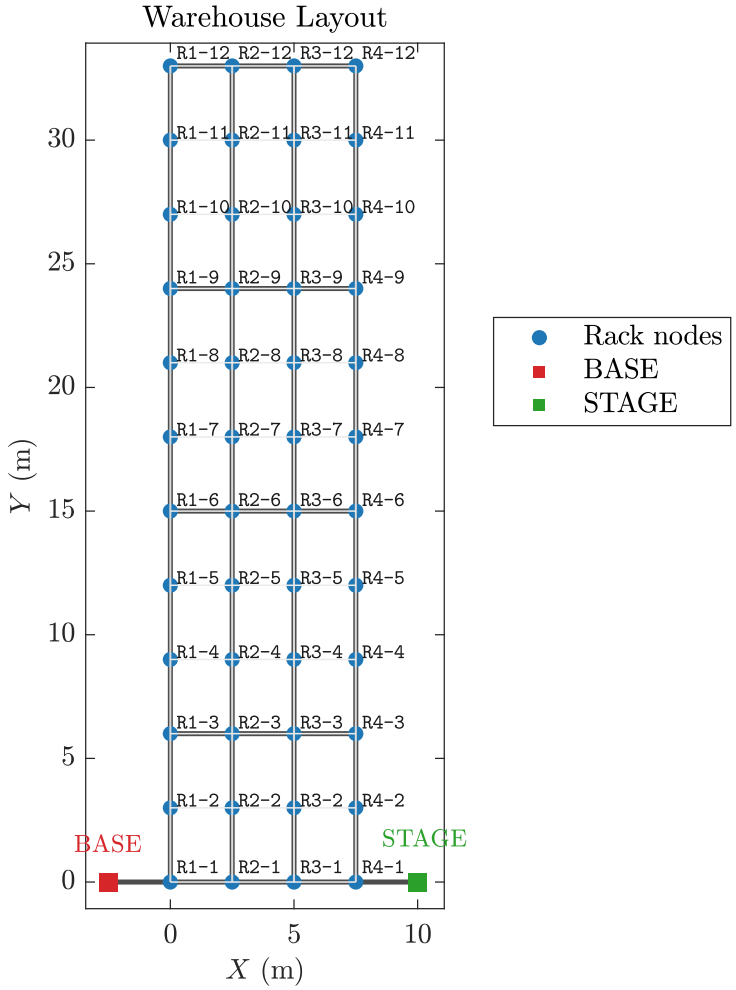


Fig. 1: Warehouse layout schematic

The heuristic $h(\cdot)$ is a Euclidean time lower bound on the remaining travel, which is feasible and consistent under nonnegative edge times and constant speed; consequently, A⋆ returns an optimal path for each fixed start–goal subpath [31], [35], [36]. This formulation ensures optimality for each subpath of a prescribed rack sequence; the multi-order route is then the sequence of subpath-optimal A⋆ paths.

2.3. Hierarchical Pathfinding A⋆ (HPA⋆)

$$\kappa(e^{\mathrm{A}}) = \min_{\pi \subseteq G \text{ inside clusters}} \sum_{e \in \pi} w(e).$$

By initiating the search on the graph G^{A}, HPA⋆ effectively reduces the global search space, enabling focused exploration within the clusters reachable via the abstract route. Recent studies in hierarchical path planning have reported significant reductions in runtime for grid and large-map problems while maintaining path quality. Additionally, hierarchical pathfinding A⋆ based on multi-scale rectangle (RHA⋆) demonstrates substantial speed improvements over A⋆, surpassing HPA⋆ while maintaining slight deviation from the optimal path cost from A⋆ [37]. Furthermore, hierarchical frameworks that incorporate topological or semantic layers achieve improved computational speed and enhanced global planning capabilities on challenging indoor maps, thereby supporting the hierarchical principle applied in HPA⋆ [38], [39]. Let the map from racks to nearest entrances be $\phi : R(o) \to R^{\mathrm{A}}(o)$. An abstract route Π^{A} that starts at BASE,

visits $R^{\mathrm{A}}(o)$, and ends at STAGE minimizes

$$\mathrm{Cost}^{\mathrm{A}}(o, \Pi^{\mathrm{A}}) = \underbrace{\sum_{e^{\mathrm{A}} \in \Pi^{\mathrm{A}}} \kappa(e^{\mathrm{A}})}_{\text{travel time}} + \underbrace{\sum_{r \in R(o)} p(r)}_{\text{picking time}}.$$

We run A⋆ on G^{A} using a heuristic function $h^{\mathrm{A}}(u) = \dfrac{\|\mathbf{x}(u) - \mathbf{x}(\mathrm{goal})\|_2}{s}$, where $h^{\mathrm{A}}(u)$ calculates the Euclidean time lower bound on remaining travel. Following this step, each abstract edge e^{A} is updated within the cluster, $\kappa(e^{\mathrm{A}})$, and concatenated to produce the low-level path π whose cycle time is defined in Eq. (2). In hierarchical planners that use grid-like warehouse layouts, the overall path length or time generally approaches optimality relative to full-graph A⋆. However, deviations in results may occur when combining each path in intra-cluster updates with precomputed crossing costs.

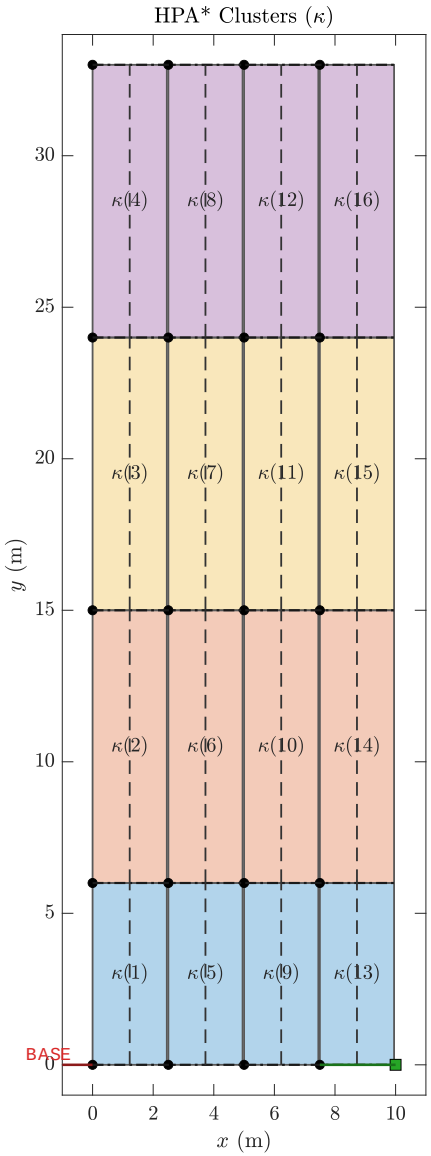


Fig. 2: HPA⋆ Clusters Patch

2.4. Multiple-Order Objective

For the set of orders $\mathcal{O} = \{o_1, \ldots, o_{m(o)}\}$, this paper considers $|\mathcal{O}| = 30$ with Order ID ORD001–ORD030 to illustrate multiple-order scenarios and seeks to minimize the total cycle time as

follows:

$$\min_{\{\Pi_o^{\mathrm{A}}\}_{o\in\mathcal{O}}} \sum_{o\in\mathcal{O}} \mathrm{Cost}^{\mathrm{A}}(o, \Pi_o^{\mathrm{A}}),$$

subject to connectivity on the graph G^{A} and coverage of the required entrance set $R^{\mathrm{A}}(o)$ for each order. In the picking order model, total order time is typically determined by the sum of travel time and picking time. The sequence of decisions or visit orders affects the travel time variable, whereas the picking time depends upon the items retrieved from the visited racks. Some studies show how storage assignment, route sequencing, and layout interact to shape cycle time and response time [32], [33], [40]. For global search, hierarchical planners optimize expansion by initially searching G^{A} and subsequently updating only within the most recent clusters. Recent studies on hierarchical A⋆ variants indicate significant reductions in runtime on grid and large map scenarios, while maintaining path quality with deviations of less than one percent compared to full-graph A⋆ [39], [41], [37]. The HPA⋆ yields near-optimal multi-order paths, achieving a significant reduction in global expansions on grid-structured layouts when integrated with rack-level picking time derived from operational data.

2.5. Complexity

The dominant computational term for HPA⋆ is

$$T_{\mathrm{HPA}^*} \approx O(|E^{\mathrm{A}}| \log |V^{\mathrm{A}}|) + \sum_{\text{inside clusters}} O(|E_K| \log |V_K|),$$

and the computational term for A⋆ in the graph view is

$$O((|E| + |V|) \log |V|). \tag{3}$$

The first term in Eq. (3) represents the global search conducted on the graph utilizing a priority queue. The summation in the second term addresses local updates limited to the clusters encountered along the path. In grid-structured and large-map warehouse layouts, initiating operations on G^{A} minimizes expansion and computational costs by reducing the node and edge sets before any intra-cluster search [41], [39]. Therefore, reducing $(|V|, |E|)$ to $(|V^{\mathrm{A}}|, |E^{\mathrm{A}}|)$ outlines practical speed-ups while maintaining cycle-time when travel time is combined with picking time. Eventually, since rack-level picking durations do not depend on the graph size, the time cost is primarily determined by the travel component of the cycle time. In contrast, the picking component remains to be determined by item retrieval at the racks visited [32], [33].

3. Results and Discussion

This study evaluates HPA⋆ within a simulated grid-structured warehouse layout comprising 50 nodes and 61 edges, as shown in Fig. 1, using 30 order IDs to compute BASE-to-STAGE routing for multiple orders picking. The warehouse layout spans 7.5 m × 33 m, with four aisles on the horizontal axis and 12 rack columns along the vertical axis. The item picker speed is 1.20 m/s, each aisle contains 12 racks, and there are 5 cross-aisle corridors.

3.1. Runtime environment

All experiments were executed on a desktop PC running Microsoft Windows 11 Education (version 24H2; OS build 26100.7462), equipped with an Intel® Core™ i7-8700 CPU (6 cores, 12 threads; base 3.20 GHz) and 16 GB RAM, 64-bit system type. MATLAB R2022b was used for all computations, plotting, and data processing.

3.2. Small Warehouse Layout Scenario

A grid-structured warehouse with a compact footprint of 7.5 m × 33 m is adopted to represent multiple-order picking and is suitable for evaluating routing and picking performance, as illustrated

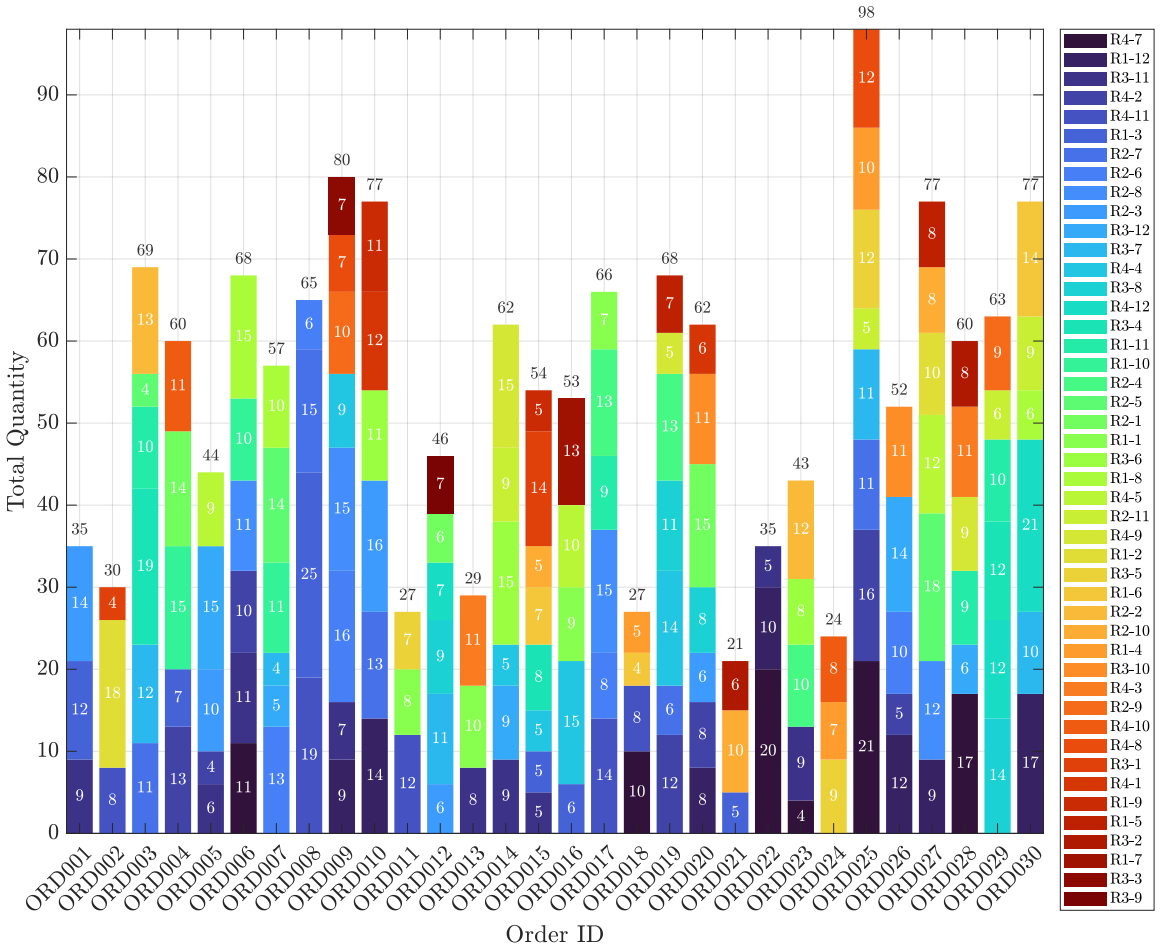


Fig. 3: Per-Order Item Quantities in a Multi-Order Scenario (ORD001–ORD030)

in Fig. 1. The global coordinate frame is defined with x along the aisles and y along the rack columns. The layout comprises four parallel aisles at $x = \{0.0, 2.5, 5.0, 7.5\}$ m and 12 rack columns at $y = 0.0$–33.0 m with 3.0 m spacing; cross-aisles are placed at columns 1/3/6/9/12, i.e., $y = \{0, 6, 15, 24, 33\}$ m. The inbound/outbound stations are BASE at $(-2.5, 0.0)$ and STAGE at $(10.0, 0.0)$. A constant picker speed $s \approx 1.20$ m/s is used to convert edge distances to travel times via $t_e = d_e/s$ using Eq. (1). The basic warehouse graph has 50 nodes and 61 edges, while the HPA⋆ hierarchical level comprises 22 nodes and 17 edges. Hierarchical edges maintain crossing costs computed via constrained local searches within their respective clusters. This approach minimizes the global search space and restricts the search to only those clusters that are directly influenced. The travel time for an order is computed by integrating the edge times along the designated path, whereas the rack-level picking time $p(r)$ is derived from order information data for each rack visited. The cycle time for each order is determined by the sum of travel time and pick time. The warehouse layout stores a total of 10315 items across all racks in four aisles, as shown in Table 1. The evaluation will focus on cycle time and the paths generated by HPA⋆. The stacked-bar chart on Fig. 3 represents the total picked quantity for each order from ORD001 to ORD030, with colored segments within each bar indicating the quantity contributed by individual rack IDs as listed in the legend.

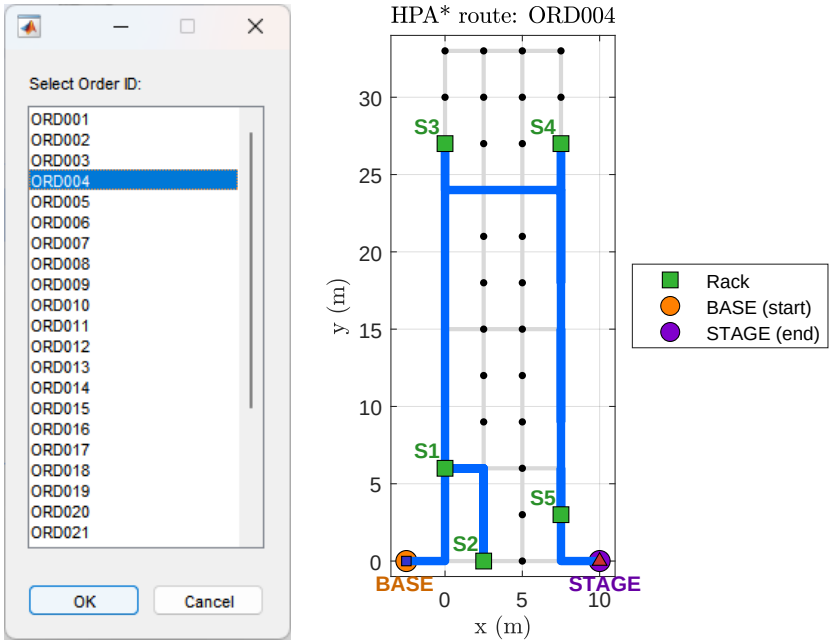
3.3. Simulation Results

In the warehouse layout as described in Subsection 3.2, search-space reduction is achieved by promoting cross-aisle connectors on columns $1, 3, 6, 9, 12$ and stations (BASE, STAGE) to entrance nodes. As a result, the global graph is reduced from $|V| = 50$, $|E| = 61$ to a hierarchical representation with $|V_a| = 22$, $|E_a| = 17$. This process eliminates 56% of the nodes and 72%

of the edges, thereby reducing the size of the A⋆ search, the number of priority-queue updates, and search-space expansion. Using a graph representation model, the hierarchical layer requires $E_a \log|V_a| = 17 \log 22 \approx 53$ logarithmic edge updates, compared with $(|E| + |V|) \log|V| = (61 + 50) \log 50 \approx 434$ on the basic graph, before local updates acquire paths within the clusters actually traversed. This indicates that the HPA⋆ global search performs roughly $434/53 \approx 8.2\times$ fewer priority-queue updates and search-space expansions before any local updates are applied inside clusters. Consequently, route computation is accelerated under the same cost model (travel time and picking time), with edge traversal times stored on the hierarchical graph.

Table 1: Total Items Stored Per Rack

Aisle 1		Aisle 2		Aisle 3		Aisle 4	
Rack ID	Quantity	Rack ID	Quantity	Rack ID	Quantity	Rack ID	Quantity
R1-1	86	R2-1	292	R3-1	301	R4-1	180
R1-2	196	R2-2	231	R3-2	146	R4-2	199
R1-3	383	R2-3	205	R3-3	94	R4-3	104
R1-4	320	R2-4	209	R3-4	142	R4-4	211
R1-5	101	R2-5	168	R3-5	322	R4-5	157
R1-6	229	R2-6	309	R3-6	245	R4-6	314
R1-7	179	R2-7	290	R3-7	112	R4-7	250
R1-8	195	R2-8	213	R3-8	228	R4-8	111
R1-9	103	R2-9	255	R3-9	32	R4-9	199
R1-10	427	R2-10	130	R3-10	169	R4-10	367
R1-11	109	R2-11	339	R3-11	69	R4-11	206
R1-12	233	R2-12	448	R3-12	276	R4-12	231
Total	2561	Total	3089	Total	2136	Total	2529
Total Items Stored: 10315 items							



(a) Select Order ID. **(b)** HPA⋆ Route for ORD004.

Fig. 4: An Example of HPA⋆ Path Planning for Single Order ID

Reducing the global search space from the raw graph ($|V| = 50$, $|E| = 61$) to the abstract graph ($|V_a| = 22$, $|E_a| = 17$) decreases the worst-case sizes of the frontier priority queue candidate and explored sets by approximately 56% (nodes: $50 \rightarrow 22$, i.e., $2.27\times$ smaller) and the edge list size by approximately 72% (edges: $61 \rightarrow 17$, i.e., $3.59\times$ smaller). In the warehouse implementation, edges represent connectors between rack locations. These reductions directly lower the number

of priority-queue computations and queue updates performed at the hierarchical-graph level. In terms of memory allocation, the HPA⋆ maintains a compact cache or lookup table of inter-cluster crossing costs on hierarchical edges, $|E_a| = 17$ entries of $\kappa(e^{\mathrm{A}})$which are reused across route queries (e.g., a multi-order picking). Consequently, global planning repeatedly leverages the same 17 cached values rather than re-expanding identical corridor details. In contrast, full-graph A⋆ repeatedly recomputes the same local subpaths for each order, creating redundant entries in the frontier priority queue and re-accessing the adjacency list. Additionally, HPA⋆'s local update reduces the allocation of search structures to only those clusters that are traversed by the hierarchical path. This approach reduces temporary memory usage to a small set of subgraphs located between cross-aisles, rather than extending it across the entire warehouse grid, while maintaining the same cost model, which includes both travel time and picking time.

Table 2: Travel Time and Picking Time For Multiple Order Cases

Order ID	Start Node	End Node	#Stops	Travel Distance (m)	Travel Time (s)	Picking Time (s)	Total Time (s)	Runtime (ms)	Success
ORD001	BASE	STAGE	3	72.5	60.417	102	162.417	1.65	✓
ORD002	BASE	STAGE	3	90.5	75.417	88.5	163.917	7.62	✓
ORD003	BASE	STAGE	6	89.5	74.583	207	281.583	130.39	✓
ORD004	BASE	STAGE	5	95.5	79.583	178	257.583	17.25	✓
ORD005	BASE	STAGE	5	96.5	80.417	131.5	211.917	19.22	✓
ORD006	BASE	STAGE	6	96.5	80.417	202.5	282.917	86.33	✓
ORD007	BASE	STAGE	6	101.5	84.583	171	255.583	119.12	✓
ORD008	BASE	STAGE	4	72.5	60.417	195.5	255.917	4.37	✓
ORD009	BASE	STAGE	8	112.5	93.75	239.5	333.25	675.45	✓
ORD010	BASE	STAGE	6	95.5	79.583	231	310.583	48.77	✓
ORD011	BASE	STAGE	3	90.5	75.417	81	156.417	4.83	✓
ORD012	BASE	STAGE	6	96.5	80.417	138	218.417	89.8	✓
ORD013	BASE	STAGE	3	84.5	70.417	87	157.417	4.43	✓
ORD014	BASE	STAGE	6	101.5	84.583	187.5	272.083	73.55	✓
ORD015	BASE	STAGE	8	96.5	80.417	158.5	238.917	634.94	✓
ORD016	BASE	STAGE	5	66.5	55.417	155.5	210.917	12.43	✓
ORD017	BASE	STAGE	6	107.5	89.583	195	284.583	105.3	✓
ORD018	BASE	STAGE	4	72.5	60.417	81.5	141.917	7.35	✓
ORD019	BASE	STAGE	7	83.5	69.583	204.5	274.083	231.31	✓
ORD020	BASE	STAGE	7	125.5	104.583	183.5	288.083	324.48	✓
ORD021	BASE	STAGE	3	72.5	60.417	61.5	121.917	3.09	✓
ORD022	BASE	STAGE	3	90.5	75.417	103.5	178.917	2.57	✓
ORD023	BASE	STAGE	5	78.5	65.417	128.5	193.917	21.22	✓
ORD024	BASE	STAGE	3	77.5	64.583	73.5	138.083	0.98	✓
ORD025	BASE	STAGE	8	101.5	84.583	290.5	375.083	584.61	✓
ORD026	BASE	STAGE	6	113.5	94.583	174	268.583	88.5	✓
ORD027	BASE	STAGE	7	114.5	95.417	230	325.417	216.15	✓
ORD028	BASE	STAGE	6	101.5	84.583	178.5	263.083	205.82	✓
ORD029	BASE	STAGE	6	113.5	94.583	189	283.583	108.94	✓
ORD030	BASE	STAGE	7	131.5	109.583	242	351.583	198.52	✓

The HPA⋆ algorithm exhibits robustness to minor layout modifications, as such changes typically affect only a limited number of hierarchical edges and their adjacent clusters. For example, using our layout scenario, closing one cross–aisle level at column 6, $y = 15\,\mathrm{m}$, removes the horizontal connectors between adjacent aisles at that level: with four aisles, this eliminates three hierarchical edges (aisle pairs 1–2, 2–3, 3–4), so the abstract graph changes from $|E_a| = 17$ to $|E_a| = 14$ while $|V_a| = 22$ remains unchanged. Consequently, only routes that traverse that level require graph-level replanning, and the recomputation is limited to the reduced hierarchical layer plus local updates inside the affected clusters; the rest of the graph and stored crossing costs are reused without modification. HPA⋆ limits the effects of a minor disturbance, specifically

a single cross-aisle closure, to a select few abstract edges and adjacent cluster updates. This approach negates the need for a complete re-evaluation of the basic graph, characterized by $|V| = 50$, and $|E| = 61$). Additionally, it upholds the travel-plus-picking time model while ensuring rapid updates to routes. Table 2 presents travel and picking times for thirty order cases between BASE and STAGE. The results indicate that all routes were successfully generated with varying runtimes depending on the number of stops. Fig. 5 illustrates the HPA⋆ path planning results for thirty order cases, showing the computed routes from BASE to STAGE. The subfigures demonstrate that the algorithm successfully generates feasible paths across varying numbers of stops and layout conditions. Table 3 provides representative path and route sequences for orders ORD002 to ORD008, showing the segment-by-segment transitions from BASE to STAGE. These examples illustrate how HPA⋆ decomposes each order into local segments while maintaining a feasible global connectivity solution.

3.4. Algorithms

Algorithm 1 extends Hierarchical Pathfinding A⋆ (HPA⋆) to support multi-order routing in a warehouse by optimizing a combined cost function that includes:

Algorithm 1 HPA⋆ routing with travel and picking time (multi-order)

Require: Warehouse graph $G = (V, E, w)$ with edge times $w(e) = d_e/s$ (picker speed s), rack-level pick times $p(r)$, set of orders $\mathcal{O} = \{o_1, \ldots, o_M\}$, stations BASE, STAGE

Ensure: Per-order low-level route π_o and cycle time $\mathrm{CT}(o)$

Build abstract level (HPA⋆)
Partition the grid into clusters bounded by cross-aisles; define entrance nodes V^{A} (cross-aisle connectors + BASE/STAGE)
Add abstract edges E^{A} (horizontal across aisles at same cross-aisle; vertical within clusters)
For each $e^{\mathrm{A}} \in E^{\mathrm{A}}$, precompute crossing cost $\kappa(e^{\mathrm{A}}) = \min_{\pi \subseteq G \text{ inside clusters}} \sum_{e \in \pi} w(e)$
for each order $o \in \mathcal{O}$ **do**
    Extract required rack visits $R(o) = \{r_1, \ldots, r_{m(o)}\}$ and map to nearest entrances $R^{\mathrm{A}}(o) = \{\phi(r_1), \ldots, \phi(r_{m(o)})\}$
    Global planning (A⋆ on hierarchical graph):
    Define $G^{\mathrm{A}} = (V^{\mathrm{A}}, E^{\mathrm{A}}, \kappa)$ and heuristic $h^{\mathrm{A}}(u) = \|\mathbf{x}(u) - \mathbf{x}(\mathrm{goal})\|_2/s$
    Run A⋆ on G^{A} for the sequence BASE $\to R^{\mathrm{A}}(o) \to$ STAGE to get subpath route Π^{A}
    Local update (cluster paths):
    For each edge traversal $e^{\mathrm{A}} = (u^{\mathrm{A}}, v^{\mathrm{A}}) \in \Pi^{\mathrm{A}}$, run a constrained search inside recent cluster $\kappa(e^{\mathrm{A}})$ to obtain low-level subpath $\pi_{u^{\mathrm{A}} \to v^{\mathrm{A}}}$ with time $\kappa(e^{\mathrm{A}})$
    Concatenate updating subpaths in order to obtain the low-level route π_o
    Cycle-time computation:
    Travel time $T_{\mathrm{travel}}(o) = \sum_{e \in \pi_o} w(e)$
    Picking time $T_{\mathrm{pick}}(o) = \sum_{r \in R(o)} p(r)$
    $\mathrm{CT}(o) = T_{\mathrm{travel}}(o) + T_{\mathrm{pick}}(o)$
end for
return $\{\pi_o, \mathrm{CT}(o)\}_{o \in \mathcal{O}}$

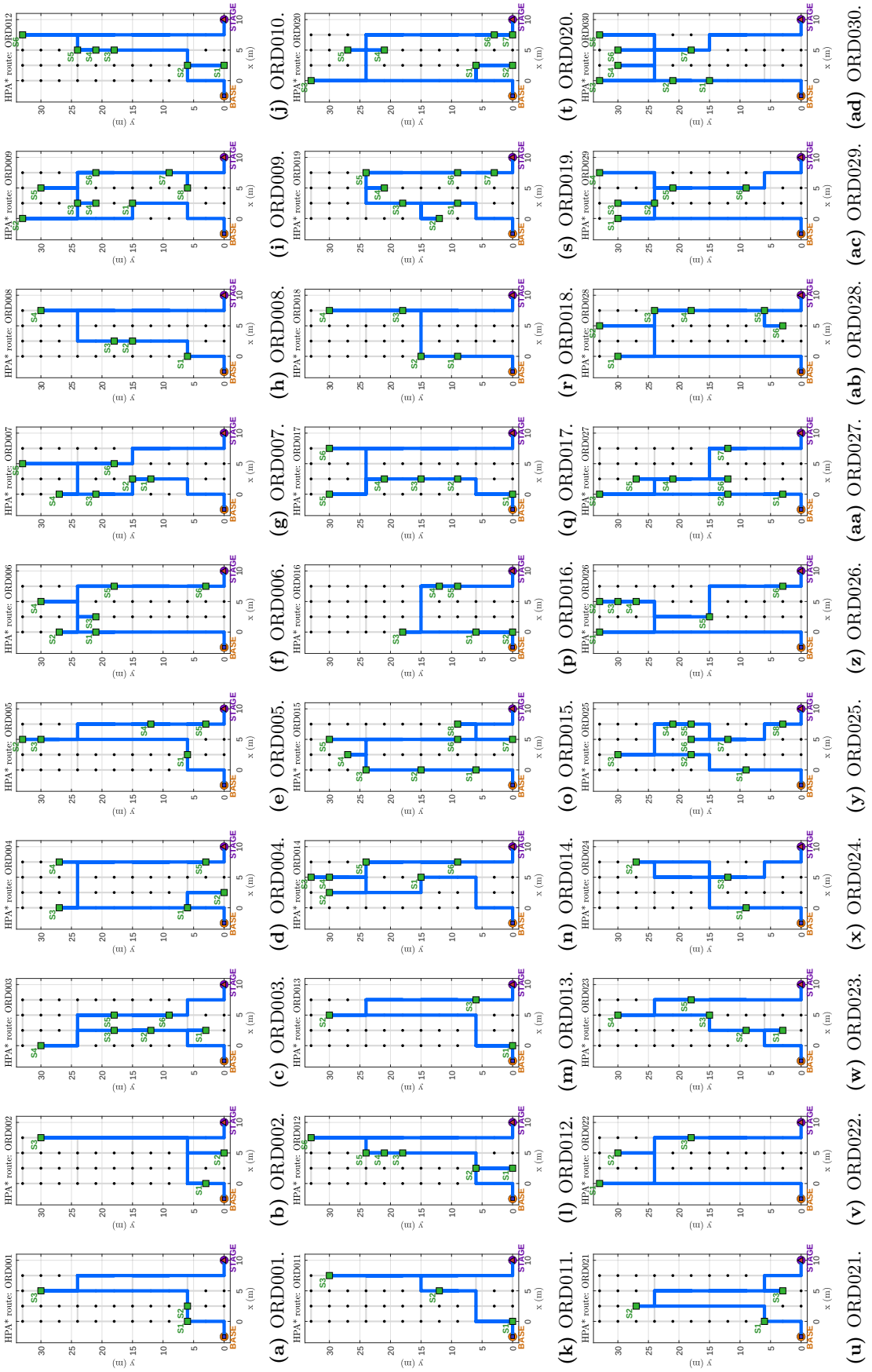


Fig. 5: HPA⋆ Path Planning Results for Multiple Orders.

Table 3: Path/Route Examples For Order ID ORD002 to ORD008

Order ID	Segment	Start Node	End Node	Path/Route
ORD002	1	BASE	R1-2	BASE → R1-1 → R1-2 → R1-3 → R1-2
	2	R1-2	R3-1	R1-2 → R1-3 → R2-3 → R3-3 → R3-2 → R3-1
	3	R3-1	R4-11	R3-1 → R3-2 → R3-3 → R4-3 → R4-4 → R4-5 → R4-6 → R4-7 → R4-8 → R4-9 → R4-10 → R4-11
	4	R4-11	STAGE	R4-11 → R4-10 → R4-9 → R4-7 → R4-8 → R4-9 → R4-7 → R4-6 → R4-4 → R4-5 → R4-6 → R4-4 → R4-3 → R4-2 → R4-1 → STAGE
ORD003	1	BASE	R2-2	BASE → R1-1 → R1-2 → R1-3 → R2-3 → R2-2
	2	R2-2	R2-5	R2-2 → R2-3 → R2-4 → R2-5 → R2-6 → R2-5
	3	R2-5	R2-7	R2-5 → R2-6 → R2-7
	4	R2-7	R1-11	R2-7 → R2-8 → R2-9 → R1-9 → R1-10 → R1-11
	5	R1-11	R3-7	R1-11 → R1-10 → R1-9 → R2-9 → R3-9 → R3-7 → R3-8 → R3-9 → R3-7
	6	R3-7	R3-4	R3-7 → R3-6 → R3-4 → R3-5 → R3-6 → R3-4
	7	R3-4	STAGE	R3-4 → R3-3 → R4-3 → R4-2 → R4-1 → STAGE
ORD004	1	BASE	R1-3	BASE → R1-1 → R1-2 → R1-3
	2	R1-3	R2-1	R1-3 → R2-3 → R2-2 → R2-1
	3	R2-1	R1-10	R2-1 → R2-2 → R2-3 → R1-3 → R1-4 → R1-5 → R1-6 → R1-7 → R1-8 → R1-9 → R1-10
	4	R1-10	R4-10	R1-10 → R1-9 → R2-9 → R3-9 → R4-9 → R4-10
	5	R4-10	R4-2	R4-10 → R4-9 → R4-7 → R4-8 → R4-9 → R4-7 → R4-6 → R4-4 → R4-5 → R4-6 → R4-4 → R4-3 → R4-2
	6	R4-2	STAGE	R4-2 → R4-3 → R4-2 → R4-1 → STAGE
ORD005	1	BASE	R2-3	BASE → R1-1 → R1-2 → R1-3 → R2-3
	2	R2-3	R3-12	R2-3 → R3-3 → R3-4 → R3-5 → R3-6 → R3-7 → R3-8 → R3-9 → R3-10 → R3-11 → R3-12
	3	R3-12	R3-11	R3-12 → R3-11 → R3-10 → R3-11
	4	R3-11	R4-5	R3-11 → R3-10 → R3-9 → R4-9 → R4-7 → R4-8 → R4-9 → R4-7 → R4-6 → R4-5
	5	R4-5	R4-2	R4-5 → R4-6 → R4-4 → R4-5 → R4-6 → R4-4 → R4-3 → R4-2
	6	R4-2	STAGE	R4-2 → R4-3 → R4-2 → R4-1 → STAGE
ORD006	1	BASE	R1-8	BASE → R1-1 → R1-2 → R1-3 → R1-4 → R1-5 → R1-6 → R1-7 → R1-8
	2	R1-8	R1-10	R1-8 → R1-7 → R1-8 → R1-9 → R1-10
	3	R1-10	R2-8	R1-10 → R1-9 → R2-9 → R2-8
	4	R2-8	R3-11	R2-8 → R2-9 → R3-9 → R3-10 → R3-11
	5	R3-11	R4-7	R3-11 → R3-10 → R3-9 → R4-9 → R4-7 → R4-8 → R4-9 → R4-7
	6	R4-7	R4-2	R4-7 → R4-6 → R4-4 → R4-5 → R4-6 → R4-4 → R4-3 → R4-2
	7	R4-2	STAGE	R4-2 → R4-3 → R4-2 → R4-1 → STAGE
ORD007	1	BASE	R2-5	BASE → R1-1 → R1-2 → R1-3 → R2-3 → R2-4 → R2-5 → R2-6 → R2-5
	2	R2-5	R2-6	R2-5 → R2-6
	3	R2-6	R1-8	R2-6 → R1-6 → R1-7 → R1-8
	4	R1-8	R1-10	R1-8 → R1-7 → R1-8 → R1-9 → R1-10
	5	R1-10	R3-12	R1-10 → R1-9 → R2-9 → R3-9 → R3-10 → R3-11 → R3-12
	6	R3-12	R3-7	R3-12 → R3-11 → R3-10 → R3-9 → R3-7 → R3-8 → R3-9 → R3-7
	7	R3-7	STAGE	R3-7 → R3-6 → R4-6 → R4-4 → R4-5 → R4-6 → R4-4 → R4-3 → R4-2 → R4-1 → STAGE
ORD008	1	BASE	R1-3	BASE → R1-1 → R1-2 → R1-3
	2	R1-3	R2-6	R1-3 → R2-3 → R2-4 → R2-5 → R2-6
	3	R2-6	R2-7	R2-6 → R2-7
	4	R2-7	R4-11	R2-7 → R2-8 → R2-9 → R3-9 → R4-9 → R4-10 → R4-11
	5	R4-11	STAGE	R4-11 → R4-10 → R4-9 → R4-7 → R4-8 → R4-9 → R4-7 → R4-6 → R4-4 → R4-5 → R4-6 → R4-4 → R4-3 → R4-2 → R4-1 → STAGE

4. Conclusion

We studied multiple order routing in a small grid-structured warehouse (7×33 m) using a combined travel-and-picking cost model. Rack-level service times were treated as ground truth from operational logs, and HPA⋆ was employed for travel-time estimation on the abstract graph. An entrance-based hierarchical system, incorporating cross-aisle connectors and stations, effectively minimized the global search space and provided fast route computation. Local updates were generated solely within the feasible paths of the clusters that were actually visited. During actual tasks, the combined cycle time, which includes both travel and picking times, remained essentially unchanged. This provides HPA⋆ as an effective and scalable framework for routing in grid-style rack systems that utilize multi-order picking.

Future research ought to investigate hierarchical configuration options, including multi-level HPA⋆, cluster size, and entrance selection, in order to regulate the trade-off between search-space reduction and path quality at larger scales. The analysis requires examining multi-order scheduling and batching by investigating sequencing methods that minimize total travel while adhering to rack-level delivery constraints. Finally, an empirical study should evaluate the robustness of these planners in dynamic operating environments, including factors like temporary aisle obstacles, speed-restricted areas, and time-window limitations.

CRediT Authorship Contribution Statement

Ig. Prasetya Dwi Wibawa: Conceptualization, Methodology, Software, Formal Analysis, Writing–Original Draft and Visualization. **Meta Kallista:** Methodology, Writing–Review & Editing. **Ramdhan Nugraha:** Validation, Writing–Review & Editing. **Heni Widayani:** Supervision, Validation. **Harish Chandra Bhandari:** Supervision, Validation, Writing–Review & Editing. **Angga Rusdinar:** Supervision, Validation.

Declaration of Generative AI and AI-assisted technologies

During the preparation of this work, the authors used Copilot in order to improve the clarity, readability, and language of the manuscript. After using this tool, the authors reviewed and edited the content as needed and take full responsibility for the content of the publication.

Declaration of Competing Interest

The authors declare no competing interests.

Funding and Acknowledgments

The authors gratefully acknowledge Research and Community Service, Telkom University's support through Research Funding Grant No. RCMS/RES/Periode 1/15425/5/2026.

Data and Code Availability

The data and code supporting the findings of this study are available from the corresponding author upon reasonable request and subject to confidentiality agreements.

References

[1] Z. Honglin, W. Yaohua, H. Jinchang, and W. Yanyan, "Collaborative optimization of task scheduling and multi-agent path planning in automated warehouses," *Complex & Intelligent Systems*, vol. 9, no. 5, pp. 5937–5948, 2023. DOI: 10.1007/s40747-023-01023-5.

[2] F. Chen, G. Xu, and Y. Wei, "Heuristic routing methods in multiple-block warehouses with ultra-narrow aisles and access restriction," *International Journal of Production Research*, vol. 57, no. 1, pp. 228–249, 2019. DOI: 10.1080/00207543.2018.1473657.

[3] K. J. Roodbergen and R. Koster, "Routing methods for warehouses with multiple cross aisles," *International Journal of Production Research*, vol. 39, no. 9, pp. 1865–1883, 2001. DOI: 10.1080/00207540110028128.

[4] Y. Zhang, M. C. Fontaine, V. Bhatt, S. Nikolaidis, and J. Li, "Multi-robot coordination and layout design for automated warehousing," in *Proceedings of the International Symposium on Combinatorial Search*, vol. 17, 2024, pp. 305–306. DOI: 10.48550/arXiv.2305.06436.

[5] A. Maoudj and A. L. Christensen, "Improved decentralized cooperative multi-agent path finding for robots with limited communication," *Swarm Intelligence*, vol. 18, no. 2, pp. 167–185, 2024. DOI: 10.1007/s11721-023-00230-7.

[6] D. Hercog, J. Marolt, P. Bencak, and T. Lerher, "Autonomous mobile robots and their integration into the order-picking process," in *Warehousing and Material Handling Systems for the Digital Industry: The New Challenges for the Digital Circular Economy*, Springer, 2024, pp. 275–308. DOI: 10.1007/978-3-031-50273-6_11.

[7] A. Papadimitriou, D. Folinas, and I. Kostavelis, "Human-robot collaborative picking system for agile warehouses," *The International Journal of Advanced Manufacturing Technology*, vol. 141, no. 7, pp. 4627–4644, 2025. DOI: 10.1007/s00170-025-16938-1.

[8] M. Zhang, E. H. Grosse, and S. Emde, "Ergonomic task allocation in an amr-assisted order picking system," *IFAC-PapersOnLine*, vol. 59, no. 10, pp. 2622–2627, 2025. DOI: 10.1016/j.ifacol.2025.09.441.

[9] Z. Zhang, Q. Guo, J. Chen, and P. Yuan, "Collision-free route planning for multiple AGVs in an automated warehouse based on collision classification," *IEEE access*, vol. 6, pp. 26 022–26 035, 2018. DOI: 10.1109/ACCESS.2018.2819199.

[10] P. Pikulin, V. Lishunov, and K. Kułakowski, "Optimizing path planning for automated guided vehicles in constrained warehouse environments: Addressing the challenges of non-rotary platforms and irregular layouts.," *Robotics*, vol. 14, no. 4, 2025. DOI: 10.3390/robotics14040039.

[11] K. Wang, W. Liang, H. Shi, J. Zhang, and Q. Wang, "Optimal time reuse strategy-based dynamic multi-AGV path planning method," *Complex & Intelligent Systems*, vol. 10, no. 5, pp. 7089–7108, 2024. DOI: 10.1007/s40747-024-01511-2.

[12] F. Nikkhoo, A. Husseinzadeh Kashan, E. Nikbakhsh, and B. Ostadi, "A bi-objective multi-warehouse multi-period order picking system under uncertainty: A benders decomposition approach," *Soft Computing*, vol. 29, no. 4, pp. 2047–2074, 2025. DOI: 10.1007/s00500-025-10495-1.

[13] P. Verma, J. M. Olm, and R. Suarez, "Traffic management of multi-agv systems by improved dynamic resource reservation," *IEEE access*, vol. 12, pp. 19 790–19 805, 2024. DOI: 10.1109/ACCESS.2024.3362293.

[14] E. Hu, J. He, and S. Shen, "A dynamic integrated scheduling method based on hierarchical planning for heterogeneous agv fleets in warehouses," *Frontiers in neurorobotics*, vol. 16, p. 1 053 067, 2023. DOI: 10.3389/fnbot.2022.1053067.

[15] W. Hönig, S. Kiesel, A. Tinka, J. W. Durham, and N. Ayanian, "Persistent and robust execution of MAPF schedules in warehouses," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1125–1131, 2019. DOI: 10.1109/LRA.2019.2894217.

[16] G. Csányi and L. Z. Varga, "Clustered reverse resumable A* algorithm for warehouse robot pathfinding," *Machines*, vol. 13, no. 12, p. 1127, 2025. DOI: 10.3390/machines13121127.

[17] K. J. Roodbergen and R. De Koster, "Routing order pickers in a warehouse with a middle aisle," *European Journal of Operational Research*, vol. 133, no. 1, pp. 32–43, 2001. DOI: 10.1016/S0377-2217(00)00177-6.

[18] N. Smolic-Rocak, S. Bogdan, Z. Kovacic, and T. Petrovic, "Time windows based dynamic routing in multi-agv systems," *IEEE Transactions on Automation Science and Engineering*, vol. 7, no. 1, pp. 151–155, 2009. DOI: 10.1109/TASE.2009.2016350.

[19] A. Botea, M. Müller, and J. Schaeffer, "Near optimal hierarchical path-finding (hpa*)," *J. Game Dev.*, vol. 1, no. 1, pp. 1–30, 2004.

[20] M. Duan, Z. Wang, X. Shao, and G. Ren, "VGA*-RRT*: A mobile robot path planning algorithm for irregular and complex maps," *IEEE Access*, 2025. DOI: 10.1109/ACCESS.2025.3552785.

[21] J. Xin, Q. Yuan, A. D'Ariano, G. Guo, Y. Liu, and Y. Zhou, "Dynamic unbalanced task allocation of warehouse AGVs using integrated adaptive large neighborhood search and Kuhn–Munkres algorithm," *Computers & Industrial Engineering*, vol. 195, p. 110 410, 2024. DOI: 10.1016/j.cie.2024.110410.

[22] Ž. Breznikar, J. Gotlih, Ž. Artič, and M. Brezočnik, "Improving AGV path planning efficiency using Genetic Algorithms with hamming distance-based initialization," *Advances in production engineering & management*, vol. 20, no. 3, pp. 299–308, 2025. DOI: 10.14743/apem2025.3.541.

[23] Y. Liu, "Global path planning method for agv of warehousing logistics based on improved ant colony algorithm," *International Journal of Industrial Engineering*, vol. 32, no. 1, 2025. DOI: 10.23055/ijietap.2025.32.1.10007.

[24] G. Sartoretti et al., "Primal: Pathfinding via reinforcement and imitation multi-agent learning," *IEEE Robotics and Automation Letters*, vol. 4, no. 3, pp. 2378–2385, 2019. DOI: 10.1109/LRA.2019.2903261.

[25] M. Damani, Z. Luo, E. Wenzel, and G. Sartoretti, "Primal _2: Pathfinding via reinforcement and imitation multi-agent learning-lifelong," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 2666–2673, 2021. DOI: 10.1109/LRA.2021.3062803.

[26] K. Okumura, M. Machida, X. Défago, and Y. Tamura, "Priority inheritance with backtracking for iterative multi-agent path finding," *Artificial Intelligence*, vol. 310, p. 103 752, 2022. DOI: 10.1016/j.artint.2022.103752.

[27] S. Bi, R. Shang, H. Luo, Y. Xu, Z. Li, and Y. Zhang, "Hac-based adaptive combined pick-up path optimization strategy for intelligent warehouse," *Intelligent Service Robotics*, vol. 17, no. 5, pp. 1031–1043, 2024. DOI: 10.1007/s11370-024-00556-z.

[28] M. Jansen and M. Buro, "HPA* enhancements," in *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, vol. 3, 2007, pp. 84–87. DOI: 10.1609/aiide.v3i1.18791.

[29] H. Antikainen, "Using the hierarchical pathfinding A* algorithm in GIS to find paths through rasters with nonuniform traversal cost," *ISPRS International Journal of Geo-Information*, vol. 2, no. 4, pp. 996–1014, 2013. DOI: 10.3390/ijgi2040996.

[30] M. Wu, E. L. M. Su, C. F. Yeong, B. Dong, W. Holderbaum, and C. Yang, "A hybrid path planning algorithm combining a* and improved ant colony optimization with dynamic window approach for enhancing energy efficiency in warehouse environments," *PeerJ Computer Science*, vol. 10, e2629, 2024. DOI: 10.7717/peerj-cs.2629.

[31] Y. Zhang, Y. Hu, J. Lu, and Z. Shi, "Research on path planning of mobile robot based on improved theta* algorithm," *Algorithms*, vol. 15, no. 12, p. 477, 2022. DOI: 10.3390/a15120477.

[32] A. Y. Alqahtani, "Improving order-picking response time at retail warehouse: A case of sugar company," *SN Applied Sciences*, vol. 5, no. 1, p. 8, 2023. DOI: 10.1007/s42452-022-05230-6.

[33] D. Loske, M. Klumpp, E. H. Grosse, T. Modica, and C. H. Glock, "Storage systems' impact on order picking time: An empirical economic analysis of flow-rack storage systems," *International Journal of Production Economics*, vol. 261, p. 108 887, 2023. DOI: 10.1016/j.ijpe.2023.108887.

[34] S. Altarazi and M. Ammouri, "Multi-criteria simulation evaluation for manual-order-picking warehouse design," in *Proceedings of the 34th European Modeling & Simulation Symposium (EMSS 2022)*, 2022, p. 010. DOI: 10.46354/i3m.2022.emss.010.

[35] A. A. Bastapure, S. S. Chiddarwar, and A. Goyal, "A comparative study of a*, rrt, and rrt* algorithm for path planning in 2d warehouse configuration space," in *International Conference on Robotics, Control, Automation and Artificial Intelligence*, Springer, 2022, pp. 95–107. DOI: 10.1007/978-981-99-4634-1_8.

[36] Y. Zheng, Z.-H. Pang, Y. Han, et al., "A* algorithm-based agvs path planning for intelligent warehousing," in *2025 Joint International Conference on Automation-Intelligence-Safety (ICAIS) & International Symposium on Autonomous Systems (ISAS)*, IEEE, 2025, pp. 1–5. DOI: 10.1109/ICAISISAS64483.2025.11051425.

[37] L. Kui and X. Yu, "A pathfinding algorithm for large-scale complex terrain environments in the field," *ISPRS International Journal of Geo-Information*, vol. 13, no. 7, p. 251, 2024. DOI: 10.3390/ijgi13070251.

[38] A. Taniguchi, S. Ito, and T. Taniguchi, "Hierarchical path planning from speech instructions with spatial concept-based topometric semantic mapping," *Frontiers in Robotics and AI*, vol. 11, p. 1 291 426, 2024. DOI: 10.3389/frobt.2024.1291426.

[39] H. Ryu, "Hierarchical path-planning for mobile robots using a skeletonization-informed rapidly exploring random tree," *Applied Sciences*, vol. 10, no. 21, p. 7846, 2020. DOI: 10.3390/app10217846.

[40] V. T. Luu, F. Chromjaková, and R. Bobák, "An optimization approach for an order-picking warehouse: An empirical case," *Journal of Competitiveness*, 2023. DOI: 10.7441/joc.2023.04.09.

[41] J. R. Sánchez-Ibáñez, C. J. Pérez-del-Pulgar, and A. García-Cerezo, "Path planning for autonomous mobile robots: A review," *Sensors*, vol. 21, no. 23, p. 7898, 2021. DOI: 10.3390/s21237898.